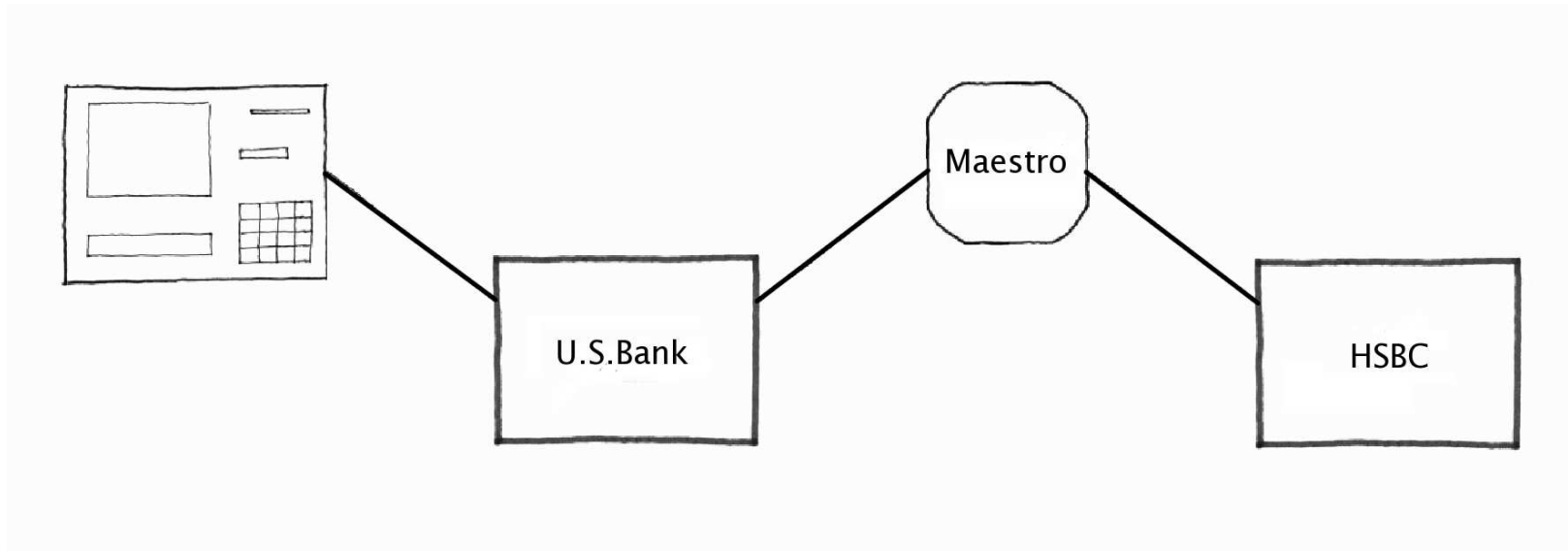

Formal Analysis of Security APIs

Graham Steel



Automated Teller Machines



Criticality of Key Management

PIN derived by:

Write account number (PAN) as 0000AAAAAAAAAAAA

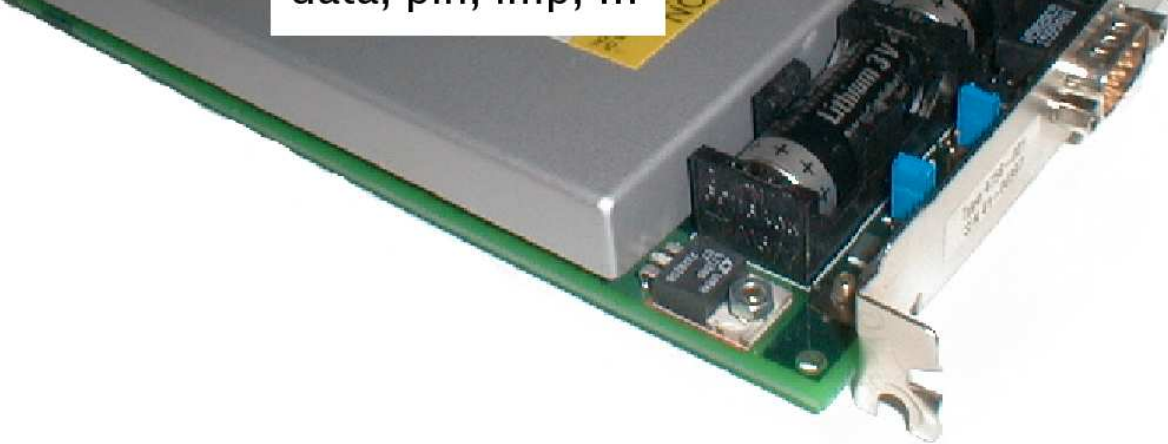
3DES encrypt under a PDK (PIN Derivation Key)

$\text{PIN} = \text{IPIN} + \text{Offset (modulo 10 each digit)}$

Offset NOT secure!

data, pin, imp, ...

—



data, pin, imp, ...

$$\{ d1 \}_{km \oplus data}$$

$$\{ pdk1 \}_{km \oplus pin}$$

CCA API - Examples

Encrypt Data:

Host → HSM : $\{ d1 \}_{km \oplus data}$, message

HSM → Host : $\{ message \}_{d1}$

Verify PIN:

Host → HSM : $\{ PINBlock \}_{p1}$, PAN, $\{ pdk1 \}_{km \oplus pin}$,

OFFSET, $\{ p1 \}_{km \oplus ipinenc}$

HSM → Host : yes/no

Importing Key Parts

‘Separation of duty’

Key $k = k_1 \oplus k_2$

Host \rightarrow HSM : k_1, TYPE

HSM \rightarrow Host : $\{ k_1 \}_{k_m \oplus k_p \oplus \text{TYPE}}$

Host \rightarrow HSM : $\{ k_1 \}_{k_m \oplus k_p \oplus \text{TYPE}}, k_2, \text{TYPE}$

HSM \rightarrow Host : $\{ k_1 \oplus k_2 \}_{k_m \oplus \text{TYPE}}$

Usually used to import a ‘key encrypting key’ ($\{ \text{KEK} \}_{k_m \oplus \text{imp}}$)

Importing Encrypted Keys

Exported from another 4758 encrypted under $KEK \oplus TYPE$

Key Import:

Host \rightarrow HSM : $\{ KEY1 \}_{KEK \oplus TYPE}, TYPE, \{ KEK \}_{km \oplus imp}$

HSM \rightarrow Host : $\{ KEY1 \}_{km \oplus TYPE}$

Attack (Bond, 2001) (part 1)

PIN derivation key: $\{ \text{pdk} \}_{\text{kek} \oplus \text{pin}}$

Have key part $\{ \text{kek} \oplus k_2 \}_{\text{km} \oplus \text{imp} \oplus \text{kp}}$ for known k_2

Host \rightarrow HSM : $\{ \text{kek} \oplus k_2 \}_{\text{km} \oplus \text{kp} \oplus \text{imp}}, k_2 \oplus \text{pin} \oplus \text{data}, \text{imp}$

HSM \rightarrow Host : $\{ \text{kek} \oplus \text{pin} \oplus \text{data} \}_{\text{km} \oplus \text{imp}}$

Attack (Bond, 2001) (part 2)

Key Import

Host → HSM : $\{ \text{pdk} \}_{\text{kek} \oplus \text{pin}}, \text{data}, \{ \text{kek} \oplus \text{pin} \oplus \text{data} \}_{\text{km} \oplus \text{imp}}$

HSM → Host : $\{ \text{pdk} \}_{\text{km} \oplus \text{data}}$

Encrypt data

Host → HSM : $\{ \text{pdk} \}_{\text{km} \oplus \text{data}}, \text{pan}$

HSM → Host : $\{ \text{pan} \}_{\text{pdk}} (= \text{PIN!})$

Formal Modelling

The 4758 can be considered 'stateless'

Key management API consists of 8(ish) commands

Model as 8 two-step protocols

Have used:

- daTac and CL-AtSe – to find attacks
- Ad-Hoc decision procedure – for verification, based on theoretical results in Cortier & Steel, FCS-ARSPA '06.

Theorem Proving with datac

One clause for each command

Host \rightarrow HSM : $\{ d1 \}_{km \oplus data}, message$

HSM \rightarrow Host : $\{ message \}_{d1}$

$$P(Msg) \wedge P(crypt(km \oplus data, D1)) \Rightarrow P(crypt(D1, Msg))$$

$$P(x) \wedge P(y) \rightarrow P(x \oplus y)$$

Associative and Commutative, Self-Inverse ($a \oplus b \oplus a \equiv b$)

Use 'XOR Constraints'

Modelling in CL-Atse

MSc Project 2006 (Gavin Keighren)

- CL-Atse is a security protocol analysis tool
- Suitable because supports XOR unification
- Model one 'role' for each command

Rediscovered known attacks on old version, and a new attack on the revised version!

Decision Procedure

First prove theoretical result showing decidability

Encode terms as integers

$$\begin{array}{cccccccc} \text{kek} \oplus \text{pin} \oplus \text{data} & \rightarrow & \text{km} & \text{kp} & \text{kek} & \text{imp} & \text{exp} & \text{data} & \text{pin} \\ 19 & \leftarrow & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}$$

Each rule is a simple operation on integers

$$\text{e.g. } x_1, x_2 \rightarrow x_1 \oplus x_2 \quad \equiv \quad n, m \rightarrow n \oplus m$$

Implemented in C, a few hundred lines

Verifies all fixes in ~ 100 s each

PIN Blocks

64-bit strings to encode a guessed PIN for encryption

e.g.

VISA format 3

PPPPFFFFFFFFFFFFFF

ISO 9564 format 0

04PPPPFFFFFFFFFFFF

0000AAAAAAAAAAAA

PIN Block Operations

Verify_Offset_PIN

Verify_IBM_PIN

Verify_PVV

Translate_Encrypted_PIN

Clear_PIN_Encrypt

Customer chooses which ones are enabled at installation time

ISO-0 Reformatting attack

(Clulow, 2003)

```
04PPPPFFFFFFFFFFFF
```

```
0000AAAAAAAAAAAA
```

Error check ($0 \leq P \leq 9$) leaks information

Extend:

Masquerade ISO-0 as VISA format 3

```
0604PPPPFFFFFFFFFFFF
```

Can now uniquely determine digits

Dectab Attacks

Controls decimalisation of $\{ \text{pan} \}_{\text{pdk}}$

Cleartext input to verify commands

Standard

0123456789ABCDEF

0123456789012345

Attack 1

0123456789ABCDEF

1123456789112345

Alter offset to establish position of digits (Bond + Zieliński, 2003)

Formal Modelling

Take a customer configuration and an API spec. as input

Using CLP, generate tree of all possible attacks

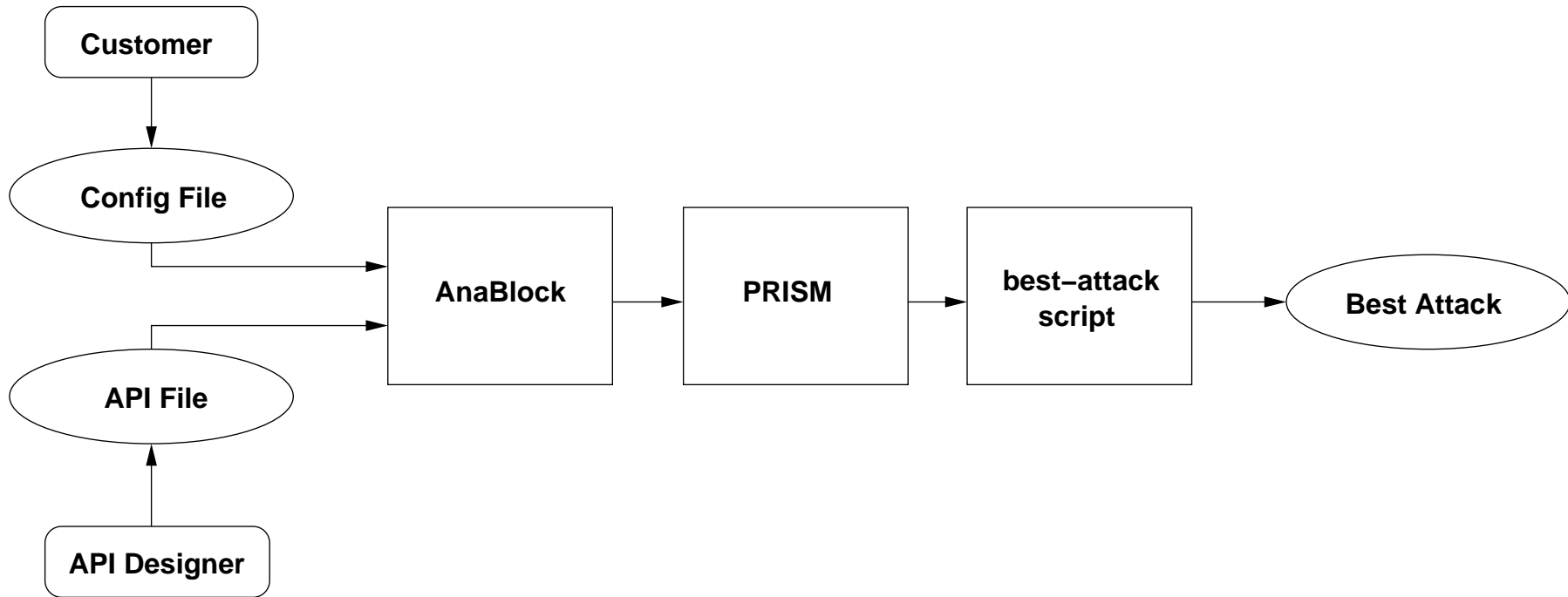
Meta-logical predicates allow us to calculate transition probabilities

Apply PRISM (Kwiatkowska et. al, 2004)

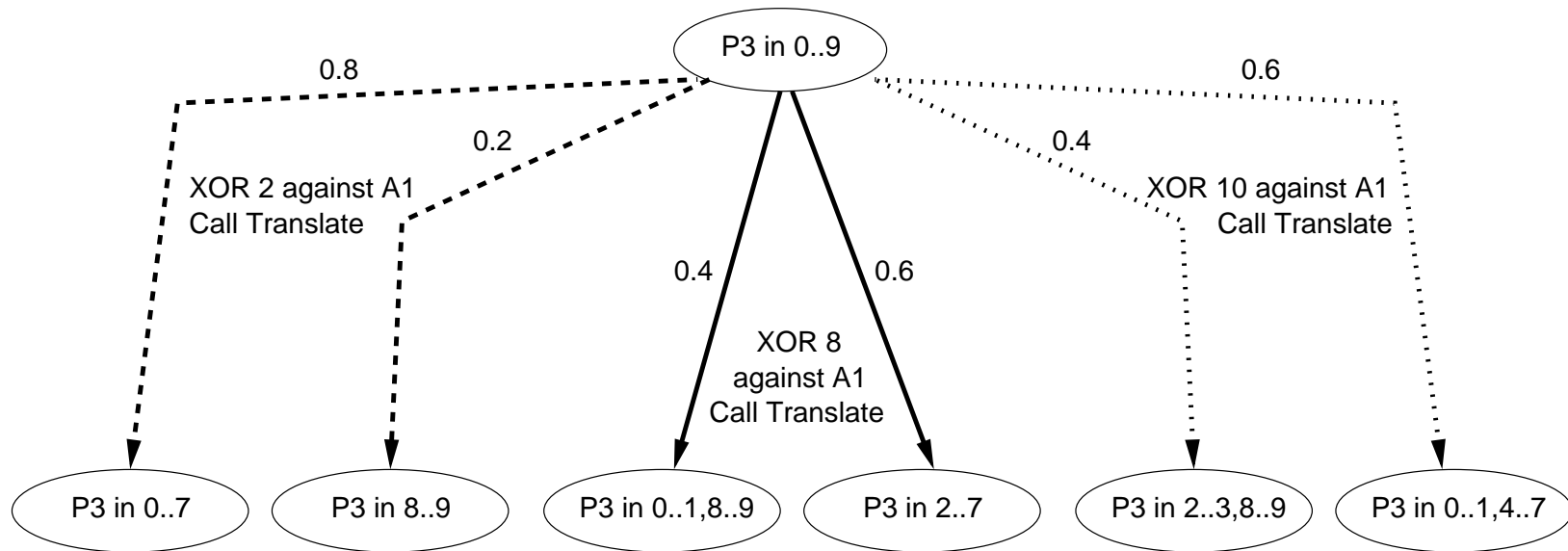
Get minimum expected number of steps to determine PIN

Generate tree for best attack

AnaBlock



Attack Trees



PCTL Properties

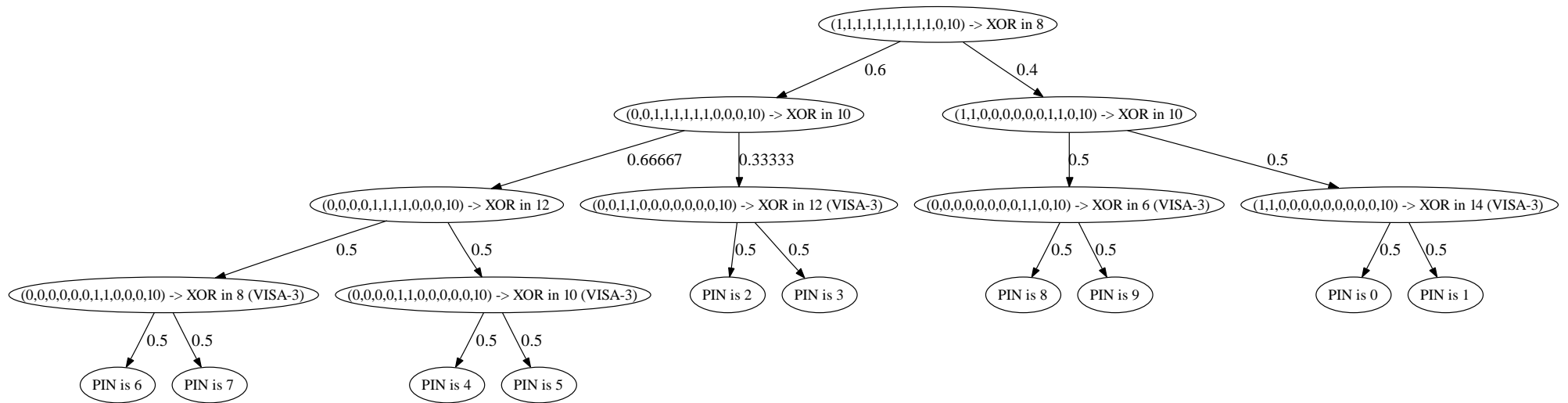
What chance of obtaining a PIN?

$$Pmax =? \quad [true \ U \ (\ P1_guessed \wedge P2_guessed \wedge \\ P3_guessed \wedge P4_guessed)]$$

What's the best way to do it?

$$Rmin =? \quad [F(\ P1_guessed \wedge P2_guessed \wedge \\ P3_guessed \wedge P4_guessed) \{ "init" \} \{ min \}]$$

Optimised Attack

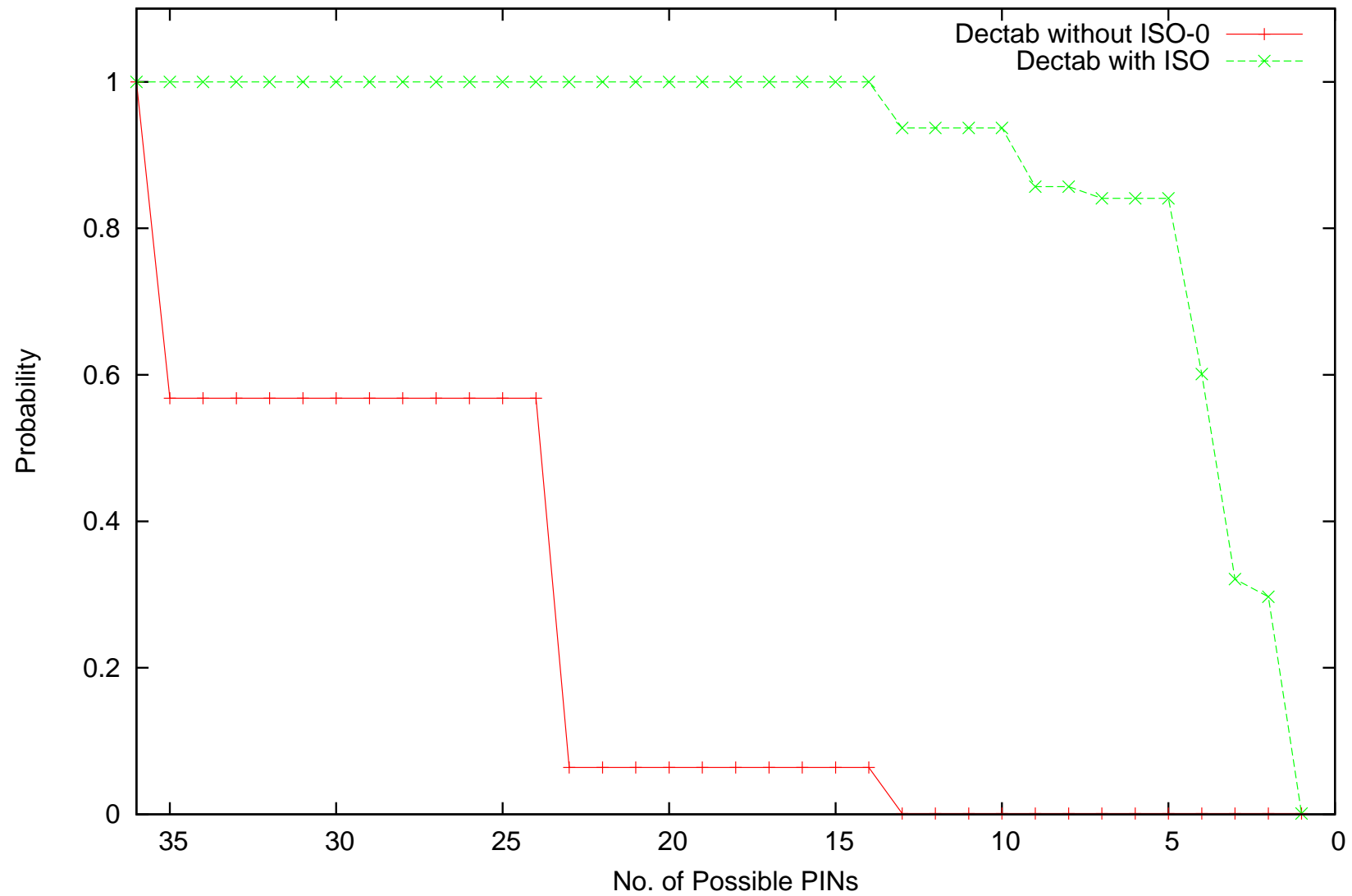


Results from Analysis (Generic API)

No.	Attack	$P(determined)$	$E(Steps)$
(1)	ISO-0 (extended)	1	13.6
(2)	Dectab	1	16.145
(3)	Dectab & ISO (restricted)	1	15.275

No.	Attack	$k = 400$	$k = 36$	$k = 24$	$k = 14$	$k = 1$
(4)	ISO-0 (restricted)	1	0	0	0	0
(5)	Dectab no offset	1	1	0.568	0.064	0.001
(6)	Dectab no offset & ISO-0 (restricted)	1	1	1	1	0.001

Performance of Dectab attack without offset



Some Open Problems

Key management APIs:

- Safe abstractions for key conjuring/explicit decryption/key generation
- Parallel key search attacks
- ‘Stateful’ APIs

PIN Processing:

- Specification language for PIN operations
- Symmetry breaking
- EMV APIs

Summary

- API analysis emerging as an important and exciting area
Used also in smartcards, DRM (TPM), automotive,...
- Have had some good results
Verification of revised CCA API
Framework for PIN Block attack analysis
- Lots of opportunities for student projects!

EPSRC Project “Automated Analysis of Security Critical Systems”

<http://dream.inf.ed.ac.uk/projects/aascs/>